Inteligencia Artificial 2017-2018

Práctica 4: Torneo de Heurísticas

GRUPO: 2213

CELIA SAN GREGORIO MORENO

ÁLVARO MARTÍNEZ MORALES

Índice

Pregunta C1

A lo largo de la práctica 4 se han implementado varias heurísticas diferentes y, aunque cada una tiene su propia aproximación al problema de la cuantificación del tablero, muchas son el resultado de realizar variaciones sobre la implementación de una heurística previa.

Las distintas versiones entregadas de cada una de las heurísticas simplemente cambiaban la ponderación de los componentes del tablero, nunca la aproximación de la cuantificación en sí.

Por último, cabe destacar que aquí figuran las heurísticas que han tenido un verdadero planteamiento y trabajo de elaboración detrás; algunas de las heurísticas subidas durante los primeros días no se ven reflejadas en este documento puesto que eran simples pruebas del resultado de heurísticas sin ningún tipo de complejidad.

Petacabras

Esta fue nuestra primera implementación de heurística, y su objetivo era crear un bot que jugara defensivamente teniendo en cuenta la diferencia de puntuación frente al jugador contrario.

Así, lo primero que tuvimos en cuenta era, en caso de que la partida hubiera terminado en el estado evaluado, si el jugador actual era el ganador. En ese caso la heurística evaluaría el estado como (a efectos prácticos) infinito, y, si era el perdedor, lo evaluaría como menos infinito.

Seguidamente, comprobaba si el estado le permitía saltar turno, en cuyo caso evaluaría un tablero como un valor de infinito ligeramente menor al de partida ganada (a efectos prácticos sería infinito también, pero manteniendo una prevalencia sobre el estado de victoria).

Por último, si no se daba ninguna de esas situaciones en el estado evaluado, la función ponderaría, en primera instancia (y con un valor superior) la diferencia de puntos entre el Kalaha propio sobre el contrario, dando prioridad a estados en los que la puntuación definitiva del jugador actual fuera mayor que la del contrario.

En segunda instancia, evaluaría **negativamente** mediante una función exponencial la cantidad de semillas en cada uno de los hoyos propios (excluyendo el Kalaha). Esto se ha hecho de manera que el bot procurase acumular la menor cantidad posible de semillas en cada uno de sus hoyos, para dejar el menor número posible de semillas expuestas a robos. Precisamente por eso se ha hecho uso de una función exponencial frente a una más simple de carácter lineal; la cantidad de semillas que se exponen a ser robadas no son las mismas si hay 2 semillas en 3 hoyos que 3 semillas en 2 hoyos. En un caso el jugador contrario puede robar como máximo 2 y en el otro 3, aunque en el fondo haya la misma cantidad de semillas en nuestros hoyos.

Por último cabe destacar que sobre la mitad del periodo de prácticas decidimos incluir el caso de empate como caso de derrota (previamente estaba dispuesto al contrario) para no atribuirle la misma evaluación que una victoria.

Las distintas versiones tendieron hacia una heurística que ponderaba más negativamente acumular semillas en los últimos hoyos (más a la derecha) lo que daba prioridad a realizar jugadas en esa área de nuestro campo.

Petabuses

El objetivo de esta heurística fue crear un bot defensivo que valorase positivamente algunos factores y muy negativamente otros.

En primer lugar, la heurística evaluaba si el juego había terminado. En caso de derrota o empate, ponderaría muy negativamente sobre el tablero (se escogió un número muy grande para simular el valor infinito). En caso de victoria, ponderaría muy positivamente.

Si el juego no había terminado, realizaba una suma con estos parámetros:

a) La resta de las fichas propias en el Kalaha con las fichas del contrario en su Kalaha. Cuantas más fichas tuviese el jugador actual en el Kalaha (siendo ese número de fichas mayor que el del contrario), la heurística ponderaría más favorablemente esa diferencia positiva.

   Si en la disposición del tablero el jugador actual tenía menos fichas en su Kalaha que el contrario, la heurística ponderaría negativamente.

   Con esta resta pretendíamos priorizar la acumulación de fichas en el Kalaha propio, ya que no pueden ser capturadas y permiten asegurar la puntuación.

b) La suma conjunta de unas operaciones de multiplicación. Estas multiplicaciones se realizaban con el hoyo propio, el hoyo opuesto y un valor numérico arbitrario.

   El objetivo de las multiplicaciones era ponderar un hoyo propio teniendo en cuenta el número de fichas del hoyo opuesto. Cuantas más fichas tuviese el jugador contrario en sus hoyos, más posibilidades habría de capturar estas fichas si en los hoyos propios hubiera 0 fichas.

Send_Nodes

La heurística de este bot surgió como una modificación de la función de evaluación del bot Petabuses. Las ponderaciones sobre el fin del juego, la posibilidad de pasar turno y el número de fichas en el Kalaha propio y el del contrario eran idénticas (salvo ligeros cambios en los valores numéricos).

La particularidad de esta heurística se encontraba a la hora de ponderar los hoyos. Si en el tablero actual algún hoyo del contrario tenía 0 fichas, la función de evaluación ponderaría negativamente, ya que no habría posibilidad de captura directamente (y el jugador contrario podría capturar las fichas del actual).

En cambio, si el hoyo del contrario tenía 1 ficha o más, la heurística se servía de las ponderaciones utilizadas previamente en el bot Petabuses para determinar cuán recomendable resultaba mover una ficha en función del número de fichas en los hoyos del contrario.

Minuano

Esta heurística surgió a partir de un intento por evaluar el tablero en función de este parámetro: la posibilidad que tenía el bot de conseguir un turno más en función del número de fichas en sus hoyos.

De nuevo, nos basamos en la heurística del bot Petabuses. Ello puede apreciarse en la evaluación del fin de partida, la situación en la que el bot puede jugar una vez más y el número de fichas en los Kalahas propio y contrario.

A todos estos parámetros se añadió la diferencia de puntuación entre el jugador actual y el contrario. La heurística ponderaría positivamente cuanta más puntuación tuviese el jugador actual respecto al contrario, y negativamente en caso contrario.

Además, si en alguno de los hoyos propios había fichas suficientes para sembrar hasta el Kalaha propio, la heurística ponderaría muy positivamente. Sino, ponderaría cuán recomendable sería escoger un hoyo para realizar la siembra con valores arbitrarios.

Cabe destacar que este bot no obtuvo muy buena posición en el torneo, ya que en la mayoría de los casos posiblemente escogiese la ponderación muy positiva si se cumplía la condición de poder pasar turno en alguno de los hoyos.

Y seguramente esa decisión le impidiera considerar otras estrategias más interesantes.

Jetstream-Sam

La función de evaluación de este bot surgió a raíz de un intento por configurar una heurística distinta a las anteriores, que valorase negativamente el hecho de pasar turno (porque no sabíamos con certeza si el jugador actual era el que pasaba turno o no) y el número de fichas en el hoyo actual en función del hoyo opuesto.

En cuanto a las ponderaciones sobre el número de fichas, tuvimos en cuenta dos factores:

a) Que el hoyo contrario tuviese 0 fichas.
b) Que el hoyo actual tuviese un número grande de fichas (5 o más).

A la diferencia del número de fichas en los Kalahas, calculado de igual forma que en los bots anteriores, restábamos unas multiplicaciones de los hoyos propios y los opuestos. Estas multiplicaciones tenían la estructura:

$$hoyo\_propio * \frac{120}{hoyo\_contrario + 1}$$

Cuantas más fichas tuviese el hoyo propio y cuantas menos fichas tuviese el hoyo contrario, la operación devolvería un valor negativo más grande. Y, por tanto, sería menos recomendable mover y acumular fichas en los hoyos propios si en los opuestos había muy pocas fichas o incluso cero.

Este bot tampoco obtuvo una buena posición en el torneo. Posiblemente, al no aplicar la regla de la captura por avaricia (si en un hoyo había 5 fichas o más), sus valoraciones sobre el tablero no fueron relevantes.

Lyssa

Esta variación de **Petacabras** incluía, sobre lo anteriormente expuesto en la aproximación de la heurística, una evaluación sobre el caso de salto de turno, el cual ya no ponderaría simplemente un valor infinito por debajo del estado de victoria, si no que se evaluaría en dicho caso sumando a nuestra puntuación la mayor cantidad de semillas en un hoyo contrario expuesta a robo (es decir, con 0 semillas en la casilla opuesta del campo propio). De esta forma calcularíamos las ganancias potenciales del salto, y evitaríamos saltar en situaciones en las que no saltando, pero tomando otra jugada que derivaría en un robo, ganaríamos muchas más semillas que en el caso de salto.

Este bot derrotaba con creces a todo lo que habíamos implementado hasta la fecha. Desafortunadamente también, sin explicación aparente, tenía una misteriosa tendencia a abandonar en situaciones fortuitas pero deterministas (es decir, sin ningún tipo de explicación, pero siempre contra determinados bots) por un probable bug en la implementación del juego, lo que le ha impedido alcanzar resultados mucho mejores que su predecesor, pues estos abandonos cuentan como derrotas para nuestro bot. De ahí su nombre, *Lyssa*, la diosa de griega de la ira, la furia y el frenesí (y en este caso, de *ragequit*).

Maniae

Esta última heurística fue, al igual que *Lyssa,* basada en **Petacabras**, pero frente a los sofisticados cambios de la anterior, ésta simplemente dejó de contemplar la posibilidad de salto en su heurística.

Sorprendentemente, no quedaba en absoluto lejos de su hermana, aunque en ésta nunca encontramos casos del bug.

Pregunta C2

Probablemente esto se deba a que, como la implementación de Negamax considera que el oponente tiene la misma heurística que el jugador actual, el jugador Bueno se reservaría de tomar ciertas buenas jugadas en base a consideraciones a las que no conseguía alcanzar su hermano jugador Regular, que seguía escogiendo las jugadas que consideraba mejores sin esa clase de miramientos.

Eso nos pasó, por ejemplo, con la mayoría de nuestras heurísticas que jugaban contra last-opt. Este bot no ganaba porque jugase mejor que las nuestros, sino porque forzaba a jugar de una manera defensiva a nuestras heurísticas, de forma que tendían a distribuir fichas por su campo y alejarlas de las posiciones libres (y con posibilidad de robo) del campo de last-opt; aunque éste en realidad nunca tendría interés en robar fichas mediante esos hoyos vacíos. Sin embargo, nuestras heurísticas no sabían eso; considerando que jugaban contra ellas mismas, acababan por perder contra algo infinitamente más simple.

Por tanto, esta solución reside en pequeños cambios que daban al bot un enfoque de juego menos defensivo y más directo, incluso más simple. Sin embargo, esta sería una mala estrategia a la larga, puesto que llevaría a tener peores resultados contra heurísticas más avanzadas.